PVCamNET

.NET ASSEMBLY FOR PVCAM

Ver. 1.0.101

Contents

Introduction

PVCamNET is a helper library for PVCam API. The main purpose of this helper is to allow Photometrics customers to write control applications for PMQI cameras in high level languages such as C#, VS.NET, LabVIEW or MATLAB.

From a technical point of view, PVCamNET is a managed DLL or .NET Assembly (by more modern vocabulary) therefore any language, that is capable of loading managed DLL, can make use of PVCamNET. The library is not only a simple one-to-one wrapper around PVCam public functions. One of the targets of PVCamNET is to simplify the complexity of development for PVCam cameras. The memory management and necessary parallelisms are already handled by the library, allowing developers to focus more on the front-end side of their application.

This text will mostly cover basics of how the library works and how to achieve most common developers goals. All public classes will be described in detail with examples to get the general idea of how the library is intended to be used. It is recommended that the latest PVCam SDK available with PVCam 3.7.5 is used, and also the required minimum for the 1.0.100.x version of PVCamNET at the time this document was written.

Public API and Types

Classes visible to the user can be divided into two categories, Public API and Types. Public classes are found in PVCamNET namespace. Types are found in PVCamNET.Types namespace.

The library contains five public classes. Library, Camera, CameraSettings, Frame and AcqBuffer. End-user applications will mostly be working with these classes. Classes use specific property types which are defined inside the PVCamNET.Types namespace, such as PmEnumItem or PmStringParam (and many others described below).

Integration

The PVCamNET library provides an object-oriented approach to PVCam library itself, which uses the C interface. The library can be used with any platform capable of loading managed DLL (assembly). Technologies of .NET Framework platform using languages such as C#, VB.NET and C++/CLI will provide the best experience when using the library.

Other non .NET related platforms such as LabVIEW, MATLAB or Python provide tools to load .NET assemblies (like PVCamNET.dll). The library cannot be used with native C++ applications since these applications can use PVCam directly. This tutorial will only describe a generic approach to the library features, meaning there will be no explanation of how to access the library functions from the specific platform. Please consult related documentation for given language to obtain this knowledge.

1 Disclaimer

Even though the PVCamNET library is trying to simplify the access to PVCam, it is in no way fool proof and does not handle all the corner cases. Knowledge of the PVCam manual is mandatory for the user/developer to successfully integrate their own solution.

2 Using the library

The library uses several public classes, most of which have internal constructors. In most cases user/developer should not be forced to create an instance of any class and pass it into the library itself. The library takes care of the lifespan of every object.

2.1 Library class

Library is a singleton class, providing the basic information about the version of the PVCam, PVCamNET, bit version library version in use and etc. The main feature of the Library class is to be an entry point into the PVCamNET library. This class provides a list of references to all currently available cameras.

The list of cameras is loaded when Library class is initialized. To update the list, the user must call either the ReloadCameras() function or the LoadCameras() function.When the application is closing the Release() function must be called to properly clean up all pre-allocated resources. On some platforms (LabVIEW) not calling Release() might cause crashes when VI execution finishes on other platforms.

2.2 Accessing camera

Getting a reference for a Camera class object can be done in two ways. The first way is to use the index to access the object in Library::Cameras collection. The second way is to use the camera name

and get the object with Library::GetCamera(string name) function. Both ways will work, but note that index in Cameras collection might not correspond with the actual camera handle!

2.3 Basics with camera

For any feature of Camera class to work it is mandatory to call Camera::Open function first. Camera class contains multiple sub-classes which serve specific purposes, namely the CameraSettings class and AcqBuffer class. Camera class itself is used to control the camera, and offers functions to start and stop the acquisition. All the camera settings such as exposure, number of frames to capture and all the parameters are kept inside the CameraSettings class. AcqBuffer class contains all the data from the camera acquisition.

2.3.1 Starting the acquisition

When Open function is called, the camera is ready to start the acquisition right away. However, it is recommended to set up an exposure time for the captured image to have any visible information. This can be done by setting the Exposure property in the CameraSettings class.

Then calling the Camera::StartLive() will start the circular buffer acquisition. Calling the Camera::StartSequence will capture a specified number of frames (In our case a single frame, which can be changed in CameraSettings class with FramesToCapture property, default value is 1). These calls are non-blocking calls, meaning functions will get out of scope, but the acquisition may be running in the background. To stop the acquisition of either Live or Sequence mode, call the Camera::Stop() function.

2.3.2 Getting camera status

Since the functions of starting acquisition are non-blocking, the camera status needs to be provided. Camera class offers a property status which reports one of two states, Busy or Idle. This status always correctly reports whether acquisition is running or not. The property supports databinding, meaning a custom event handler can be connected to this property. This handler will get fired when the property changes (for specifics see our PVCamNET examples).

2.3.3 Accessing captured frames

After acquisition, only the frames in the buffer are stored in the memory. For the sequence acquisition, the buffer is set to contain all frames set up by the FramesToCapture property. For the live acquisition only, the last X frames from the acquisition are available, where X is the size of circular buffer.
Frame is represented by the object of Frame class. Each frame in the buffer can be accessed by setting the AcqBuffer::BufferIndex property to a desired index and then accessing the selected frame via the FrameToDisplay property.

2.3.4 Displaying image

PVCamNET supports displaying the image with a built in feature called CamDisplay. This feature allows the user to see the image in a new window. This is quite handy with console applications and LabVIEW. To display the image for the given camera, call the Library::ShowImage(camObject) function. CamDisplay has some built-in functionalities, such as zooming, min/max intensity scaling and region drawing.

2.3.5 Setting region

There are two ways to set the region on a camera. The first way is to add Types::PmRegion object inside the Camera::Settings::Regions collection (If collection is empty camera is capturing full sensor). The second way is using CamDisplay where the region can be added or modified by mouse. Both options are synchronized by databinding using the same model (Camera::Settings::Regions collection).

2.4 Working with parameters

Functionality and details of the individual parameters are already described in the PVCam manual. PVCamNET provides object oriented access to the individual parameters. PVCamNET provides the PmParam property for each parameter with correct types. Because the types are fundamentally different, iterating through all the parameters with their types assigned was not a viable option. If there is a need to iterate through all parameters CameraSettings::Parameters, the hash map is available.

However, this hash map only provides the user with PmParamBase classes, which then needs to be casted to the proper type. PmParamBase contains the RawType property which is an int value reported by PVCam specifying the type. The type definitions can be found in pvcam.h (search for TYPE_INT16 to get the first definition). Comparing these values with RawType properties will enable the user to cast PmParamBase to the proper type (i.e. PmNumericParam<unsigned short>). The parameter browser is provided as sample code.

2.5 Saving the image

There are multiple ways to save the image in .NET framework, however there is no built-in support for TIFF image formatting. PVCamNET expands on this using the TiffLib.NET helper library. With this library present, PVCamNET can save images to the TIFF format.. One way is to save the n th image with AcqBuffer::SaveAstiff(nTh), and another way is to save a stack of images from n th to m th image with AcqBuffer::SaveAsTiff(startIndex, endIndex) and the last way is to save a stack of all the images in the buffer with SaveAsTiff(). Note that if fewer frames are captured than the capacity of the Buffer only the actual number of captured frames will be saved and multi-tiff supports up to 2GB of image data. To save images into other formats, the Frame::RawData property provides raw image data which can be used by any .NET framework method for saving images.

2.6 Data binding

It is important for the application to listen to the PropertyChanged event available in every class to ensure an application always has the valid value from the property. PropertyChangedEventArgs provides the PropertyName property to get the string name of the property which is being updated. Please see the sample code for more information.

2.7 Error handling

PVCamNET function calls may fail with an exception. It is recommended to catch for both System::Exception (general errors) and PVCamNET::Types::PmException (PVCam specific errors). Also, it is recommended to listen to the AsyncError event in the Camera class. The event is fired when an error happens on the acquisition thread, which would normally be uncatchable by the application (since there is no place to put a try/catch block).

API Classes

3 Library

This static class enables user to get basic information about the system, cameras, and PVCam.

3.1 Functions

3.1.1 GetCamera

Parameters	String cameraName
Return type	Camera
Specifier	-
Main scenario	Returns an camera object given the camera name
Alt. scenario	Exception
Exception	PmException – Camera was not found

3.1.2 ShowImage

Parameters	Camera cam
Return type	void
Specifier	-
Main scenario	Opens camera display for a given camera
Alt. scenario	-
Exception	-

3.1.3 ReloadCameras

Parameters	void
Return type	void
Specifier	-
Main scenario	PVCam gets reloaded and the Cameras collection is updated
Alt. scenario	-
Exception	PmException – when any PVCam calls fails

3.1.4 LoadCameras

Parameters	void
Return type	void
Specifier	-
Main scenario	PVCam gets reloaded and the Cameras collection is updated
Alt. scenario	-
Exception	PmException – when any PVCam calls fails

3.1.5 Release

Parameters	void
Return type	void

Specifier	-
Main scenario	Releases all resources manually, when the library is released it cannot be opened again during the same processes runtime. Call this function only when the application is closing.
Alt. scenario	-
Exception	PmException – when any PVCam function calls fail.

3.2 Properties

3.2.1 IsReleased

Type	Bool
Accessibility	Static read only
Data binding	No
Description	Returns true if Library was already released, meaning it cannot be opened again.

3.2.2 Cameras

Type	PmReadOnlyObservableCollection<Camera>
Accessibility	Read only
Data binding	Yes
Description	Returns a list of camera class objects, representing connected cameras in the system during the load time of the library. Note that even though Camera objects can be accessed directly via the index, this Index might not equal the actual camera handle. Using the GetCamera function is recommended.

3.2.3 PVCamVersion

Type	String
Accessibility	Read only
Data binding	No
Description	Returns PVCam version in string in format X.X.X

3.2.4 BitVersion

Type	String
Accessibility	Read only
Data binding	No
Description	Returns PVCamNET bit version in string in format Assembly: XXbit

3.2.5 Version

Type	String
Accessibility	Read only
Data binding	No
Description	Returns PVCamNET version in string in format X.X.X

4 Camera

Objects of this class represents actual camera connected to the system. Every operation with a camera device is done via this class.

4.1 Functions

4.1.1 Open

Parameters	-
Return type	void
Specifier	-
Main scenario	Camera is not open - PVCam call pl_cam_open(..) - Loads all parameters - Selects gain index to 0
Alt. scenario	Camera is already open - PmException
Exception	PmException – any PVCam call fails Exception – native memory allocation fails

4.1.2 Close

Parameters	-
Return type	void
Specifier	-
Main scenario	If acquisition is running - Acquisition gets stopped PVCam calls pl_camera_close(..) If camera display is opened - Display is closed
Alt. scenario	-
Exception	PmException – any PVCam call fails

4.1.3 StartSequence

Parameters	-
Return type	void
Specifier	-
Main scenario	Starts sequence acquisition process on a parallel thread. Sets camera status to Busy.
Alt. scenario	-
Exception	PmException – any PVCam call fails

	Exception – native memory allocation fails

4.1.4 StartLive

Parameters	-
Return type	void
Specifier	-
Main scenario	Starts live acquisition process on a parallel thread. Sets camera status to Busy.
Alt. scenario	-
Exception	PmException – any PVCam call fails Exception – native memory allocation fails

4.1.5 Stop

Parameters	-
Return type	void
Specifier	-
Main scenario	Stops the running acquisition. Sets camera status to Busy.
Alt. scenario	If no acquisition is running the function does not do anything.
Exception	PmException – any PVCam call fails Exception – native memory allocation fails

4.2 Properties

4.2.1 AcquisitionBuffer

Type	AcqBuffer
Accessibility	Read only
Data binding	No
Description	Manages frames captured in the acquisition.

4.2.2 Settings

Type	CameraSettings
Accessibility	Read only
Data binding	No
Description	Keeps current settings of the camera.

4.2.3 Status

Type	CameraStatus
Accessibility	Read only
Data binding	No
Description	Returns current camera status.

4.2.4 IsOpen

Type	bool
Accessibility	Read only
Data binding	No
Description	Reports if camera is opened.

4.2.5 Name

Type	string
Accessibility	Read only
Data binding	No
Description	Reports name of the camera.

4.2.6 FrameFact

Type	FrameFactory
Accessibility	Read only
Data binding	No
Description	Returns a frame factory instance.

4.3 Events

4.3.1 AsyncError

Type	Action<PmException>
Fire event	When an exception is fired in acquisition thread.

5 CameraSettings

This class contains all the options and parameters supported by the camera including the camera handle. Note that only commonly used properties are listed below as there are over 70 properties.

5.1 Properties

5.1.1 Handle

Type	int
Accessibility	Read only
Data binding	No
Description	Reports the camera handle given by PVCam

5.1.2 ExposureTime

Type	unsigned int
Accessibility	Read / Write
Data binding	Yes
Description	Exposure time for acquisition

5.1.3 FramesToCapture

Type	unsigned short
Accessibility	Read / Write
Data binding	Yes
Description	Sets a number of frames to be captured by sequence acquisition

5.1.4 SensorWidth

Type	PmNumericParam<int>
Accessibility	Read only
Data binding	Yes
Description	Returns full width of the camera sensor

5.1.5 SensorHeight

Type	PmNumericParam<int>
Accessibility	Read only
Data binding	Yes
Description	Returns full height of the camera sensor

5.1.6 BitDepth

Type	PmNumericParam<int>
Accessibility	Read only
Data binding	Yes
Description	Returns current bit depth based on selected port / speed combination

5.1.7 ExposureResolution

Type	PmEnumParam
Accessibility	Read only
Data binding	Yes
Description	Holds the supported options for the parameters and current values for setup

5.1.8 XBinning

Type	Unsigned short
Accessibility	Read / Write
Data binding	Yes
Description	Set and gets the X binning.

5.1.9 YBinning

Type	Unsigned short
Accessibility	Read / Write
Data binding	Yes
Description	Set and gets the Y binning.

5.1.10 SupportedBinning

Type	PmReadOnlyObservableCollection
Accessibility	Read
Data binding	Yes
Description	Get the supported binning.

5.1.11 Clear Cycles

Type	Unsigned int
Accessibility	Read / Write
Data binding	Yes
Description	Set and gets the clear cycles

5.1.12 ClearingModes

Type	PmEnumParam
Accessibility	Read / Write
Data binding	Yes
Description	Set and gets the clear mode

5.1.13 ExposureMode

Type	PmEnumParam
Accessibility	Read / Write
Data binding	Yes
Description	Set and gets the exposure mode.

5.1.14 ExposureOutMode

Type	PmEnumParam
Accessibility	Read / Write
Data binding	Yes
Description	Set and gets the exposure out mode.

5.1.15 SpeedTable

Type	Unsigned short
Accessibility	Read / Write
Data binding	Yes
Description	Set and gets the Y binning.

5.1.16 Regions

Type	PmSpeedTable
Accessibility	Read / Write
Data binding	Yes
Description	Set and gets the speed table.

6 AcqBuffer

This class holds all data regarding the current/last acquisition. With the start of each acquisition object is cleared to the default state. This class also provides events for incoming new frames and frame limiter.

6.1 Properties

6.1.1 BufferIndex

Type	Unsigned int
Accessibility	Read / Write
Data binding	No
Description	Sets and gets the current index of secondary buffer. Based on this value the FrameToDisplay property is updated.

6.1.2 BufferSize

Type	Unsigned int

Accessibility	Read / Write
Data binding	Yes
Description	Sets and gets the secondary buffer size.

6.1.3 CameraBufferSize

Type	Unsigned int
Accessibility	Read / Write
Data binding	Yes
Description	Sets or gets the current circular buffer size of acquisition buffer in frames.

6.1.4 SingleFrameSize

Type	Unsigned int
Accessibility	Read only
Data binding	Yes
Description	Returns a single frame size in bytes. The value changes based on selected ROI and Binning.

6.1.5 Fps

Type	Float
Accessibility	Read only
Data binding	Yes
Description	Reports current fps during the acquisition.

6.1.6 FpsDisp

Type	Float
Accessibility	Read only
Data binding	Yes
Description	Reports current display fps during the acquisition.

6.1.7 LatestFrameUpdateFps

Type	Unsigned int
Accessibility	Read / Write
Data binding	Yes
Description	Sets or gets the target fps for frame limiter.

6.1.8 DoCapturing

Type	bool
Accessibility	Read / Write
Data binding	Yes
Description	Enables or disables disk streaming.

6.1.9 CapturedFrames

Type	Unsigned int
Accessibility	Read only
Data binding	Yes
Description	Returns the number of captured frames in the acquisition.

6.1.10 DroppedFrames

Type	Unsigned int
Accessibility	Read only
Data binding	Yes
Description	Returns the number of dropped frames in the acquisition.

6.1.11 CacheUsage

Type	Unsigned int
Accessibility	Read only
Data binding	Yes
Description	Returns the usage of fast acquisition cache.

6.1.12 FrameToDisplay

Type	Unsigned int
Accessibility	Read only
Data binding	Yes
Description	Returns the number of captured frames in the acquisition.

6.1.13 UseScaling

Type	Boolean
Accessibility	Write only
Data binding	Yes
Description	This must be enabled to calculate min, max, mean and histogram.

7 Frame

This class represents a single frame in the managed memory. It provides basic information about the image. It also provides the Bitmap property which can be displayed directly via WPF controls or CamDisplay.

7.1 Properties

7.1.1 FrameNumber

Type	Unsigned int
Accessibility	Read only
Data binding	No
Description	Returns frame number

7.1.2 BofTimeStamp

Type	double
Accessibility	Read only
Data binding	No
Description	Returns BOF time stamp

7.1.3 EofTimeStamp

Type	double
Accessibility	Read only
Data binding	No
Description	Returns EOF time stamp

7.1.4 Height

Type	Unsigned int
Accessibility	Read only
Data binding	No
Description	Returns captured height of the image/region

7.1.5 Min

Type	Unsigned int
Accessibility	Read only
Data binding	No
Description	Returns minimum brightness in the image

7.1.6 Max

Type	Unsigned int
Accessibility	Read only
Data binding	No
Description	Returns maximum brightness in the image

7.1.7 MeanValue

Type	double
Accessibility	Read only
Data binding	No
Description	Returns mean value brightness in the image

7.1.8 Bitmap

Type	BitmapSource
Accessibility	Read only
Data binding	No
Description	Returns BitmapSource object scaled to 8bits

7.1.9 BitmapScaled

Type	BitmapSource
Accessibility	Read only
Data binding	No
Description	Returns BitmapSource object scaled to 8bits with min/max scaling

7.1.10 RawImageData

Type	Array<unsigned short>
Accessibility	Read only
Data binding	No
Description	Returns raw image data array

7.1.11 Histogram

Type	Array<unsigned int>
Accessibility	Read only
Data binding	No
Description	Returns histogram data array

7.1.12 Histogram8Bit

Type	Array<unsigned int>
Accessibility	Read only
Data binding	No
Description	Returns 8 bit histogram data array

PM Types

PmTypes can found in PVCamNET.Types namespace, they are mostly used to wrap native PVCam types. All PmTypes support databinding via the OnPropertyChanged event.

8 PmEnumItem

This type provides value-name pair container.

8.1 Properties

- String Name
- Int Value

9 PmException

This inherits from System::Exception. When thrown, it will read the PVCam error code and error message from PVCam.

9.1 Properties

- Int PVCamErrorCode
- String PVCamErrorMessage

10 PmRegion

This type represents the rgn_type from PVCam, but it does not contain binning. The position is defined from the top-left corner of the region using the X, Y property. Width and Height are measured from this position. All properties support databinding using OnPropertyChanged event. This is the only Type (except PmException) which has public constructor and it is allocable by the host application.

10.1 Properties

- Unsigned short X
- Unsigned short Y
- Unsigned short Width
- Unsigned short Height

11 PmSpeedPort

This type represents the SpeedPort combination and it is used for PmSpeedTable. Each speed table combination has a set of Gains assigned.

11.1 Properties

- PmEnumItem Port
- Int SpeedIndex
- Int BitDepth
- Int PixTimeNs
- PmReadOnlyObservableCollection<PmEnumItem> Gains

- String Label
- PmEnumitem CurrentGain

12 PmSpeedTable

Contains already built speed table accessible via the Option property. The value to be set is to be selected into the Current property.

12.1 Properties

- PmSpeedPort Current
- PmReadOnlyObservableCollection<PmSpeedPort> Gains

13 PmObservableCollection<T>

This class is inherited from ObservableCollection, which allows limiting maximum number of items inside the collection. Also, it allows conversions to raw array type.

13.1 Functions

- ToArray() – return type array<T>

13.2 Properties

- Int MaxItems

14 PmParamBase

This class serves as a base class for specific parameter types. It provides only basic attributes shared by all PVCam parameters. All parameter types support databinding.

14.1 Properties

- Int Id
- String Name
- Bool Available
- Bool IsReadOnly

15 PmBoolParam

Wraps Boolean PVCam parameters.

15.1 Properties

- Bool Default
- Bool Current

16 PmEnumParam

Wraps enum PVCam parameters.

16.1 Properties

- PmEnumItem Default
- PmEnumItem Current
- PmReadOnlyObservableCollection<PmEnumItem> Options

17 PmNumericParam<T>

Wraps numeric PVCam parameters. T is a data type used for numeric types.

17.1 Properties

- T Default
- T Current
- T Min
- T Max
- T Step

18 PmPostProcessingParam

Wraps PP PVCam parameters.

18.1 Properties

- Unsigned int Default
- Unsigned int Current
- Unsigned int Min
- Unsigned int Max
- Unsigned int Step

19 PmStringParam

Wraps string PVCam parameters.

19.1 Properties

- String Current